Natural Language Processing

Lab 7: Maximum Likelihood Estimation Models

Objectives

Students will implement N-gram language models (unigram, bigram, trigram). They will learn how to estimate word sequence probabilities using the chain rule and the Markov assumption. They will compute raw (MLE) probabilities, then apply Laplace (add-one) smoothing and Interpolated Kneser-Ney smoothing to handle unseen events. Finally, they will evaluate model performance using perplexity and generate example sentences by sampling from their model.

N-Grams and Maximum Likelihood Estimation (MLE)

N-Gram is an algorithm based on statistical language model. Its basic idea is that the contents (phonemes, syllables, letters, words or base pairs) inside the text are operated by sliding window of size N according to bytes, forming a sequence of byte fragments of length N.

Here N can be 1, 2 or any other positive integer, although usually we do not consider very large N because these n-grams rarely occur in many different places.

Each byte fragment is called a gram, and the frequency of all grams is counted and filtered according to a pre-set threshold to form a list of key grams, which is the vector feature space of this text, and each kind of gram in the list is a feature vector dimension.

The goal is to compute the probability of a sentence or sequence of words.

Let's begin with the task of computing P(w|h), the probability of a word w given some history h. Suppose the history h is "its water is so transparent that" and we want to know the probability that the next word is the:

$$P(the|its\ water\ is\ so\ transparent\ that).$$

One way to estimate this probability is from relative frequency counts: take a very large corpus, count the number of times we see its water is so transparent that, and count the number of times this is followed by the. This would be answering the question "Out of the times we saw the history h, how many times was it followed by the word w", as follows:

$$P(the \mid its\ water\ is\ so\ transparent\ that) = \frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)}$$

For finding out the probability of a sequence of words:

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_1 w_2 \dots w_{i-1})$$

$$\text{P(its, water, is, so, transparent, that)}$$

$$\text{P(“its water is so transparent”)} = \text{P(its) x P(water}|\text{its) x P(is}|\text{its water) x P(so}|\text{its water is) x P(transparent}|\text{its water is so)}$$

Unigram Model

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

$$P(the \mid \text{its water is so transparent that}) \approx P(the)$$

Bigram Model

$$P(w_i|w_1 w_2 \ldots w_{i-1}) \approx \prod_i P(w_i|w_{i-1})$$

$$P(the|\text{its water is so transparent that}) \approx P(the|that)$$

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

$< s >$ I am Sam $< /s >$
$< s >$ Sam I am $< /s >$
$< s >$ I do not like green eggs and ham $< /s >$

$$\begin{array}{lll} P(I| < s >) = 2/3 & P(Sam| < s >) = 1/3 & P(am|I) = 2/3 \\ P(< /s > |Sam) = 1/2 & P(Sam|am) = 1/2 & P(do|I) = 1/3 \end{array}$$

Task 1

Create a model for Maximum Likelihood Estimation (MLE) with unigram, bigram, trigram and 50-gram segmentation on the Adventures_Holmes.txt. Compare the time taken for each inference.

Perplexity

The perplexity (sometimes called PPL for short) perplexity of a language model on a test set is the inverse probability of the test set, normalized by the number of words. For a test set $W = w_1 w_2 \ldots w_N$,:

$$\text{perplexity}(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Applying chain rule:

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

For unigrams:

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i)}}$$

For bigrams:

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Task 2

Evaluate the MLE models in Task 1 through the perplexity metric and compare them.

Laplace Smoothing

The simplest way to do smoothing is to add one to all the n-gram counts, before we normalize them into probabilities. All the counts that used to be zero will now have a count of 1, the counts of 1 will be 2, and so on. This algorithm is called Laplace smoothing.

For unigrams:

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

For bigrams:

$$P_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{\sum_w (C(w_{n-1}w) + 1)} = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

Task 3

Perform laplace smoothing on MLE models from Task 1.